

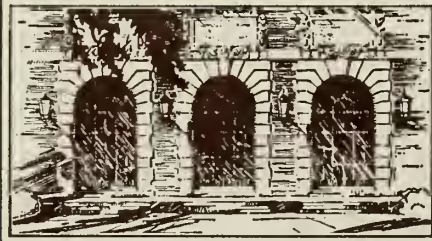
LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84

I l 6 r

no. 171-187

cop. **2**





Digitized by the Internet Archive
in 2013

<http://archive.org/details/experimentalresu184yama>

184
184
3

EXPERIMENTAL RESULTS FOR LOCAL FILTERING
OF DIGITIZED PICTURES

by

S. Yamada
J. P. Fornanago

June 15, 1965

UNIVERSITY OF ILLINOIS

AUG 19 1965

LIBRARY



DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

Report No. 184

EXPERIMENTAL RESULTS FOR LOCAL FILTERING
OF DIGITIZED PICTURES*

by

S. Yamada
J. P. Fornanago

June 15, 1965

Department of Computer Science
University of Illinois
Urbana, Illinois

* This work was supported in part by the U.S. Atomic Energy Commission under Contract No. AT911-1)-1018.

510.84
Iler
no. 184
cop. 3

TABLE OF CONTENTS

	Page
1. INTRODUCTION.	1
2. LABELING.	2
2.1 General.	2
2.2 Use.	2
2.3 Process.	3
2.4 Results.	3
3. NOISE CLEANING.	6
3.1 General.	6
3.2 Generation of Random Noise	6
3.3 Noise Cleaning Process	7
3.4 Results.	10
4. RECONNECTION.	20
4.1 General.	20
4.2 Generation of Burst Noise.	20
4.3 Reconnection Process	20
4.4 Results.	24
5. EXTRACTION OF FEATURES.	31
5.1 General.	31
5.2 Extraction Process	31
APPENDIX I.	33
APPENDIX II	34

1. INTRODUCTION

The purpose of this paper is to investigate a general plan for the local recognition of visual patterns. This local recognition is the basis of a higher level recognition, global recognition [4].

The following subjects are investigated in our local recognition level:

1. A General Labeling Routine,
2. Cleaning of Random Noise,
3. Shaping, including Burst Noise Correction, and
4. Preliminary Extraction of Feature Objects (the basis of higher level recognition).

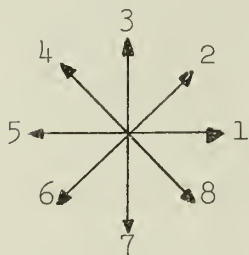
The local filtering theory follows lines developed by Yamada [1]. Programs to test the filtering theory were written in PAX [2], a parallel processing language written for the IBM 7094-1401 System at the University of Illinois. Portions have been written in SCATRE [3], the assembly language for that system. An additional routine not described in the original PAX report has been used. This routine is described in Appendix I.

-
- [1] Yamada, S., "Local Recognition of Pictorial Images," Department of Computer Science, University of Illinois, Urbana, Illinois. Report in preparation.
 - [2] Stein, J. H., "User's Manual for PAX," Department of Computer Science, University of Illinois, Urbana, Illinois. Report No. 147 (July, 1963).
 - [3] Wolf, W., "SCATRE for the IBM 7094. Library Subroutine: L1-UØI-SCRE-19-BX," Department of Computer Science, University of Illinois, Urbana, Illinois.
 - [4] Yamada, S., "Global Recognition of Pictorial Images," Department of Computer Science, University of Illinois, Urbana, Illinois. Report in preparation.

2. LABELING

2.1 General

A subroutine which is common to both programs in Parts 3 and 4 is known as LABEL. As the name implies, the routine labels the points of a picture according to the approximate direction of the line through the point. The labeling convention is as follows:



Actually, the program assigns at least two directions to each point labeled. If all lines were straight lines, only one label would be necessary (i.e., a line running east and west would need only one label (1) rather than two labels (1 and 5)). However, a point on a curved portion of a line or on a corner of a line may require two directions which are not diametrically opposed (i.e., the labeling directions may be 1 and 4).

2.2 Use

The subroutine is entered at the symbol LABEL using PAX index register 8 as a link. The picture to be labeled is in a plane called INPUT. Upon return to the calling program, INPUT is unchanged, all labeled points are in a plane called NEWVAL (new values), and all unlabeled points are in a plane called NOCON (no connections), and the labels are in eight planes beginning with IODIR (input-output directions). Plane IODIR-1+k contains all points labeled in the k direction. (Note: NOCON = NEWVAL.) Two other planes, TEMP1 and TEMP2, are used as temporary scratch planes by the subroutine. PAX index registers 1 through 4 are altered.

2.3 Process¹

The initial labeling is done in a somewhat unconventional manner. Rather than look at immediate neighbors, the program examines points two units away along the primary directions. A point is labeled in directions d and $d+4$ if the point itself and the second neighbors along the d and $d+4$ directions are all black in INPUT. The labeled points are added to NEWVAL. After all directions have been examined, NOCON is set to the complement of NEWVAL.

After this initial labeling, the IODIR planes should contain enough information about the straight portions of the input picture that the corners and gaps can easily be filled in without referring to INPUT.

The filling process is accomplished by examining immediate neighbors. A point is labeled in the d direction if its d neighbor and either its $d+3$, $d+4$, or $d+5$ neighbor has been labeled in the initial labeling. This allows points to be labeled along straight portions, or around curves or corners.

Points which were labeled in the initial labeling are not allowed to pick up additional labels in the filling. This is done by filling only points in NOCON. Note that filled points are not subtracted from NOCON until all filling is completed.

2.4 Results

This labeling algorithm has proven to be useful in two cases. The first is where the lines of a picture have a width of from one to three points and are very long compared with their width. When wider lines are used, the interior points are multiply labeled, but the border points of the line are labeled according to the direction of the border. Therefore, this routine performs point labeling, rather than line labeling.

The second case is where the lines are embedded in random noise (see Section 3).

The most prominent fault of this routine is that the lines are "chopped." Since the labeling depends on the continuity of the line through the point, the end points of the line are not labeled. In the case where noise is not prevalent, it is a simple matter to extend the labels to the ends

1. See program listing in Appendix I and flow chart in Figure 1.

of the INPUT lines. This is done in the labeling part of the reconnection routine (Part 4). However, where noise is prevalent, the extension becomes extremely difficult, and therefore was not used in the noise cleaning routine (Part 3).

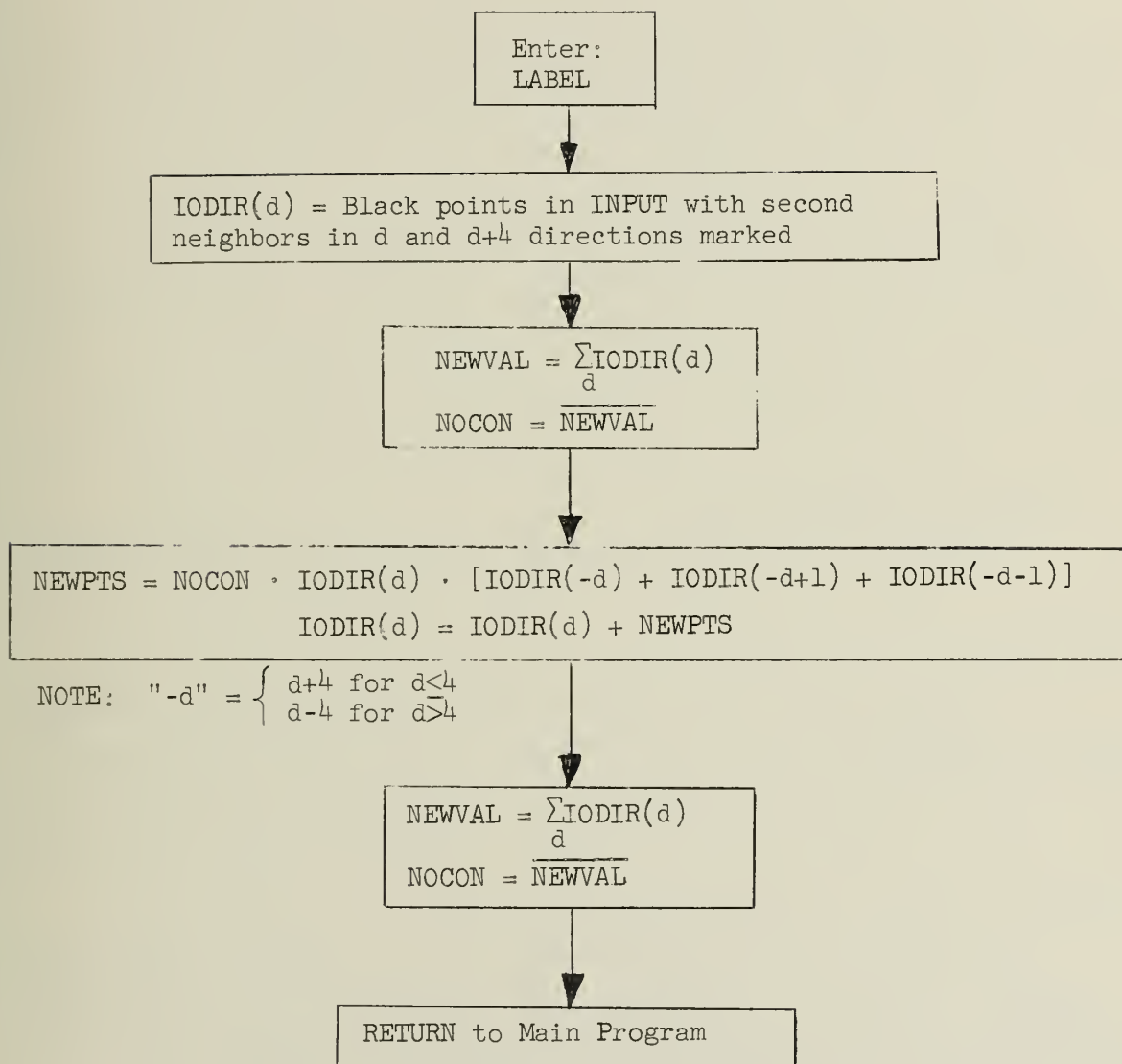


Figure 1. LABEL Subroutine Flow Chart

3.1 General

This noise cleaning routine is a simulation of a general algorithm worked out by Yamada [1] for special application to large noise cases. The large noise case was chosen since it is a more general problem. If no a priori information is known about the extent of noise in a picture, the most powerful cleaning routine must be used. On the other hand, if it is known that the noise is small, a less powerful but more efficient routine may be used.

This cleaning algorithm cannot be programmed precisely because it is neither definite enough nor simple enough to be programmed with any degree of efficiency. Also, the algorithm is based on a well-defined noise input. As will be seen in paragraph 3.2, the noise used is not well-defined.

However, the original filtering theory has been modified and realized as a programmable and fairly efficient routine. The label routine described in a previous section uses aspects of the general theory. This section describes the extension of the label routine into a noise cleaning routine.

3.2 Generation of Random Noise

A pattern was chosen as a basis of study, and noise was added to this pattern. The noise was to be of a random nature, but of uniform density over the entire plane.

In an effort to achieve this, a random number generator was used to generate random coordinates. The random generator routine was entered twice, and the two numbers obtained were used as x- and y- coordinates of a point. That point was then made "bad" by reversing its value. If the point had previously been reversed, no change was made.

The density of noise was controlled by the number of points which were changed. By using a noise level scale from 0 to 9, the entire range of possible degrees of noise was covered. A level of n indicates that, on the average, n points of each 3×3 window are "bad." In other words, in an $m \times m$ plane, $nm^2/9$ points were made bad in order to achieve a noise level of n .

The noise obtained by this generation routine is usable, but far from ideal. There are two reasons for this. First, the random number generator did not generate uniformly from 0 to 71, but put a slight emphasis on values from 0 to 56. Second, it was found that sets of two randomly generated numbers do not define random coordinates in a two-dimensional plane.

3.3 Noise Cleaning Process¹

The basis of the program is the LABEL routine. But before attempting to label a noisy picture, an initial mapping of the domain to the point is performed. That is, only points (black or white) with greater than or equal to a certain number of black neighbors (including itself) are input as black points to the label routine. This number, T, is called the threshold level.

Because of the nonuniformity of the noise, no single T-value is optimum. Using T=4, too many bad points are retained, and the picture is not thoroughly cleaned. Using T=5, not enough points are retained and small portions of lines are erased.

Therefore, the picture to be cleaned is labeled twice, using T=4 and T=5. The portions erased by using T=5 are picked up by extending the results of T=5 into the results of T=4 on a direction continuation basis. The resulting picture is then reextended into T=4 until no new points are picked up.

The extension is begun with the results of the labeling using T=4 in four planes starting with T4. T4(d) contains the points labeled in the d and d+4 directions. Similarly, T5(d) contains the d and d+4 labeled points using T=5. T5ALL contains all points labeled using T=5. T5BAR = $\overline{T5ALL}$, and contains all points not labeled.

The end points in T5 are selected by direction and are put into one of eight planes, DENDS(d). A point is an end point in the d direction if it is in T5(d), has no labels in the side or corner in the d direction, and has any label in the opposite direction (see Example 1). In an effort to retain continuity at corners, all points in T4(d+1) and T4(d-1) are added to DENDS(d) to form DENDSX(d).

1. See program listing in Appendix II and flow chart in Figure 2.

Example 1. Examples of End Point Definitions

Requirements for end points in the 2 direction:

D	N	N
D	X	N
L	D	D

Requirements for end points in the 3 direction:

N	N	N
D	X	D
D	L	D

In these examples, X is the point in question and must be labeled in the same direction as the end point is to be labeled; N is a point with No label; L is a Labeled point (regardless of direction); and D is a "Don't Care" point (does not influence the selection of end points).

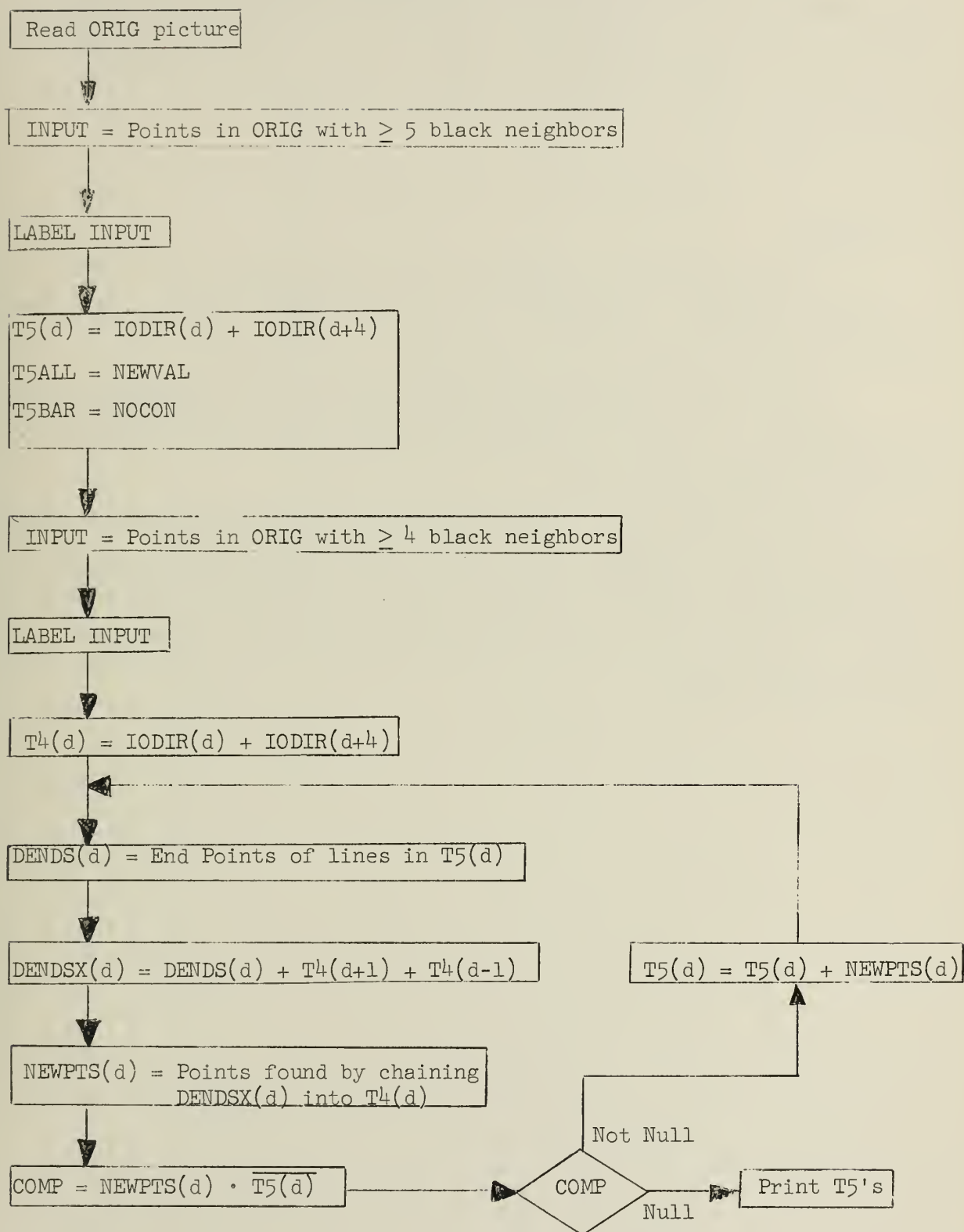


Figure 2. Noise Cleaning Flow Chart

The growing is accomplished by chaining the points in DENDSX(d) into T4(d). The new points are put into NEWPTS(d).

After all directions have been grown, NEWPTS(d) is tested against T5(d). (NEWPTS(d) may contain points previously in T5(d).) If no new points have been picked up, the process is complete and the T5's contain the final picture. If new points are found, they are added to the T5's and the extension is reiterated.

3.4 Results

The program was tested using noise levels from 0 to 3. The results are shown in Figure 3. Comparison of input and output pictures show that all prominent features of the input are retained in the output.

The routine obviously does not work well in the highest noise case, but this degree of noise is extremely rare in actual pictures produced by electrical or mechanical means.

The symbols which mark points on the processed pictures of Figure 3 specify the labels assigned to the points in accordance with the code of the table below. For example, a point on a processed picture which is marked "3" was at the juncture of a "2-6" (Northeast-Southwest) line segment and a "3-7" (North-South) segment.

Symbol on Output Picture	Direction Code				
	1-5	2-6	3-7	4-8	
E	x				} 1 direction
A		x			
N			x		
B				x	
1	x	x			} 2 directions
2	x		x		
3		x	x		
4		x		x	
6			x	x	
+	x	x	x		} 3 directions
-	x	x		x	
*	x		x	x	
.		x	x	x	
\$	x	x	x	x	4 directions



ORIGINAL PICTURE

ERROR LEVEL = 0

-13-



ORIGINAL PICTURE

ERROR LEVEL = 1

```

                                     E
                                EEEEEEEEEEEEEEEEE
                                1EE1EEE
                                A11A
                                11111
                                11111
                                A 11A
                                A
                                A
                                A
                                A
                                AA
                                AA
                                3A
                                3
                                N
                                N
                                N
                                3N
                                AN2EEEE1EEEEEEEEEE
                                1ANEEEEC
                                11+11
                                A11EE3A
                                AA1E*N3
                                AAA 6NN
                                3AA 6NN
                                33A NNN
                                N3 NNN
                                33 NNN
                                3A NNN
                                AA NNN
                                AA NNN
                                33 NN
                                3N N
                                N6 N
                                N6 N
                                NN2 NB
                                NNN B
                                N3 441 N
                                NN H4B 6
                                NN B4*6B
                                NN BB4B
                                NN 4EEEEEE1
                                NNN
                                NNN
                                N6N
                                N66
                                NN
                                NN
                                66N
                                6N
                                66
                                66
                                N
                                N
                                NN
                                N
                                N
                                NN
                                NN
                                NNN
                                NN
                                6N
                                6
                                66
                                6

                                     E
                                EEEEEEEEEEEEEEEEE
                                EE4EEE
                                EEE4444
                                4444B
                                444B
                                BB6B
                                BB6B
                                BBB
                                B

                                E44*BB
                                BB*6B
                                BB6*E
                                B6N6
                                NN3
                                NNN
                                NNN
                                NNN
                                NNN
                                NNN
                                NNN
                                3NN
                                3N
                                A+3
                                B133
                                .+3
                                NN33
                                N3A
                                3A
                                AA
                                1A
                                AA

                                1E
                                A1E
                                31-BN
                                AAA B6
                                33A 6
                                NN N6
                                N N
                                3N NN
                                N3 N6
                                3N 6
                                3NN NB
                                366 N66
                                A+E B 3NN
                                31EEE4E 3NN
                                AA
                                EEE236
                                3A E4E*66
                                3 B4*4
                                3 B4
                                N BB
                                B
                                N

```

PROCESSED PICTURE

ERROR LEVEL = 1

-16-


```

1EEEEEEEE
R      A122E      1EEE64
BBB    1111221    A      44
B44E11+1+1-6N    A      4B
B4C1113N B6      B6B
.5+1- N 6B      BBB
B6N B BB      BBB
NN B 44      B*B
1
B
N
NN3
N.3
3AB
E
B 33 BB
BB4133 B
BB 3N
666B
3.6N
EEEEEEEE
NEEEEEEEEE*44
N 4E13+*2EEEE6 AEE44**2*E
3 64-++23 B 44***
B A+1$4$++A B B66$6
BBEEEE111162$A. B 66+N
BE1+311452+AA N6+3
N6+++A NN+1EEE 333
N3+AA NN 333
N33A N6 A N3N
333 NN A NNN
A33N NN EE1 3 666
3+6R NN .66
A2+BB N3 4EE 3N*
NN3E4E N E1+33
N63 N EE41+31
3.N N 613A
5.6 N NN1A
1EBB NB N3A
BBB 6 B 3A
B6 B441 N A3A
6 BBB N A33 A
B N6 B466 E113 A A
B 3N BBB4EE1EE11AN A 3
1ENE-13N BB4411+11*B6 A 224EEEE
A N6-$+N B4+E+E 44*BA 33VN
EE*N$5$*6R 4E1 B4*$B 3$5$3N
N6+11.B B6N6B A1EE4E111E66N
BN*$+144B B4*44E1A114EE311446N
NN53 4E44A NA 633A B66
N6.3 N A3A 66B
336N B6 A33 N66
N*N 641+A NN*1
EEEEEE46 NB-33 N2*6
66 N3.. $E*6N
66 NN+N 33-$*N B
AB VN2+E 33AN6. 4EE
A1E 6 B A N A 13+E$.NN BB4EN
N 3 B N A B336A336N A 1E23
6 N 1 NA 3+21-4266 A N3
N6 NN B E1 3 11NE1E4EB44EE1 3A
N 3N 64BE1 53
N NB46 A5
N 46H AA
BA N 6611A
6 B V A
N 2E5 A
N *-5*66 N B
E44+E +EE
E44EEEE EE

```

PROCESSED PICTURE

ERROR LEVEL = 3

4.1 General

Another form of noise is burst noise. It differs from random noise because it is completely local; all points in a small area have been erased. Clearly, the noise cleaning routine cannot be used to correct this noise. The method of correction is to reconnect the lines which have been broken.

To select the lines which are to be reconnected, one must consider such things as the direction and the curvature of the lines, and the size of the gaps. It is important that the decision consider the continuity of the curvature of the lines.

4.2 Generation of Burst Noise

The pattern used in the noise cleaning testing is also used in the testing of the reconnection routine. The point where the curved line in Figure 3 crosses itself was selected for the center of the noise area.

A plane called NOISE is used as a negative mask on the input picture. Each time the program is iterated, the points in NOISE are smeared to produce a larger noise area. Then a plane called NOISY is set as $\text{NOISY} = \text{ORIG} \cdot \text{NOISE}$, thus causing points to be removed from ORIG.

4.3 Reconnection Process¹

The first step in the reconnection process is the selection of end points. This is done by nearly the same process used in the noise cleaning routine. The difference is that the labels are extended to nonlabeled points after the LABEL routine. The nonlabeled points are given the labels of their neighbors by a directional smearing process.

The end point coordinates are listed, then the program leaves the parallel processing mode and processes this list.

1. See program listing in Appendix II and flow chart in Figure 4.

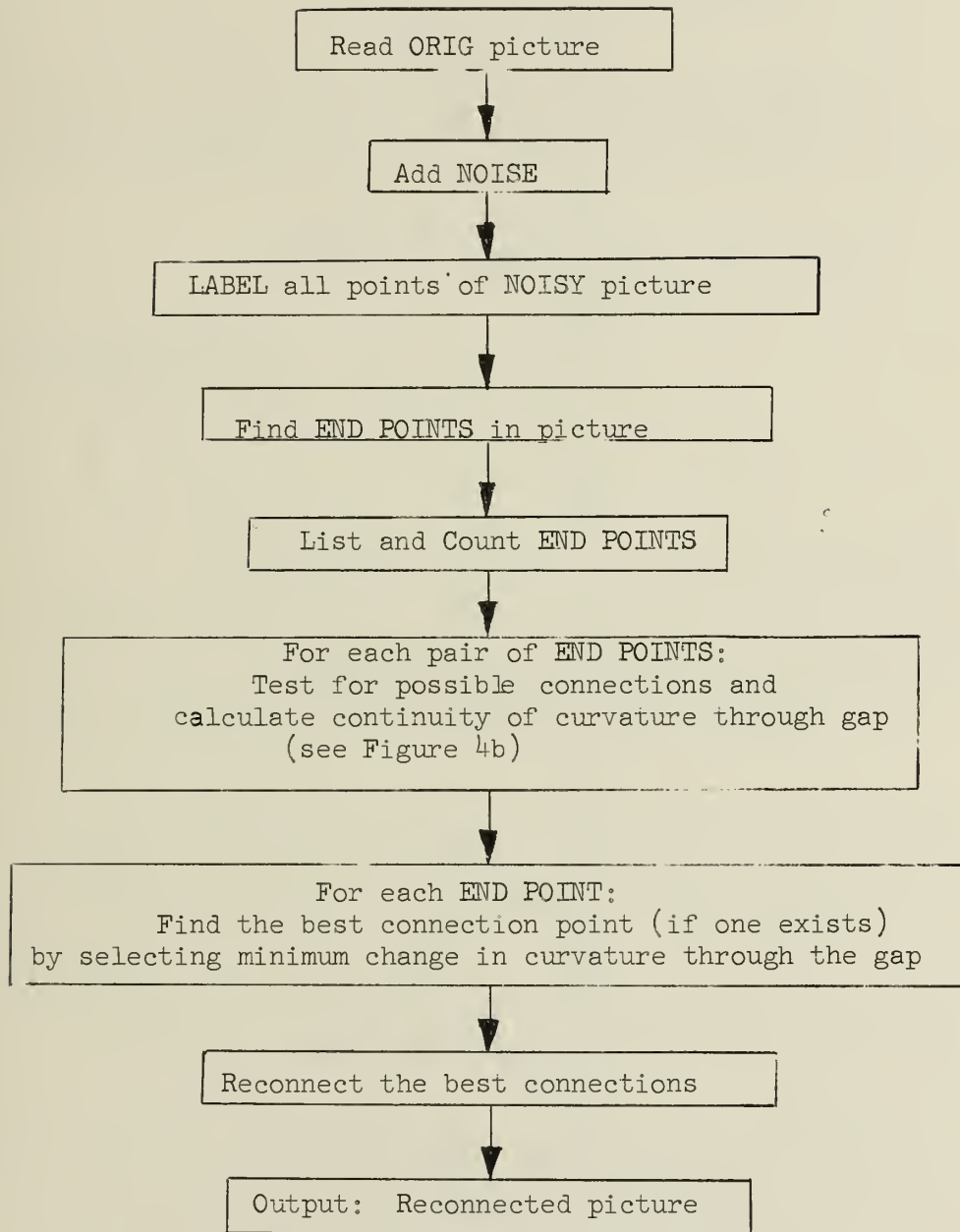


Figure 4a. Reconnection Routine Flow Chart

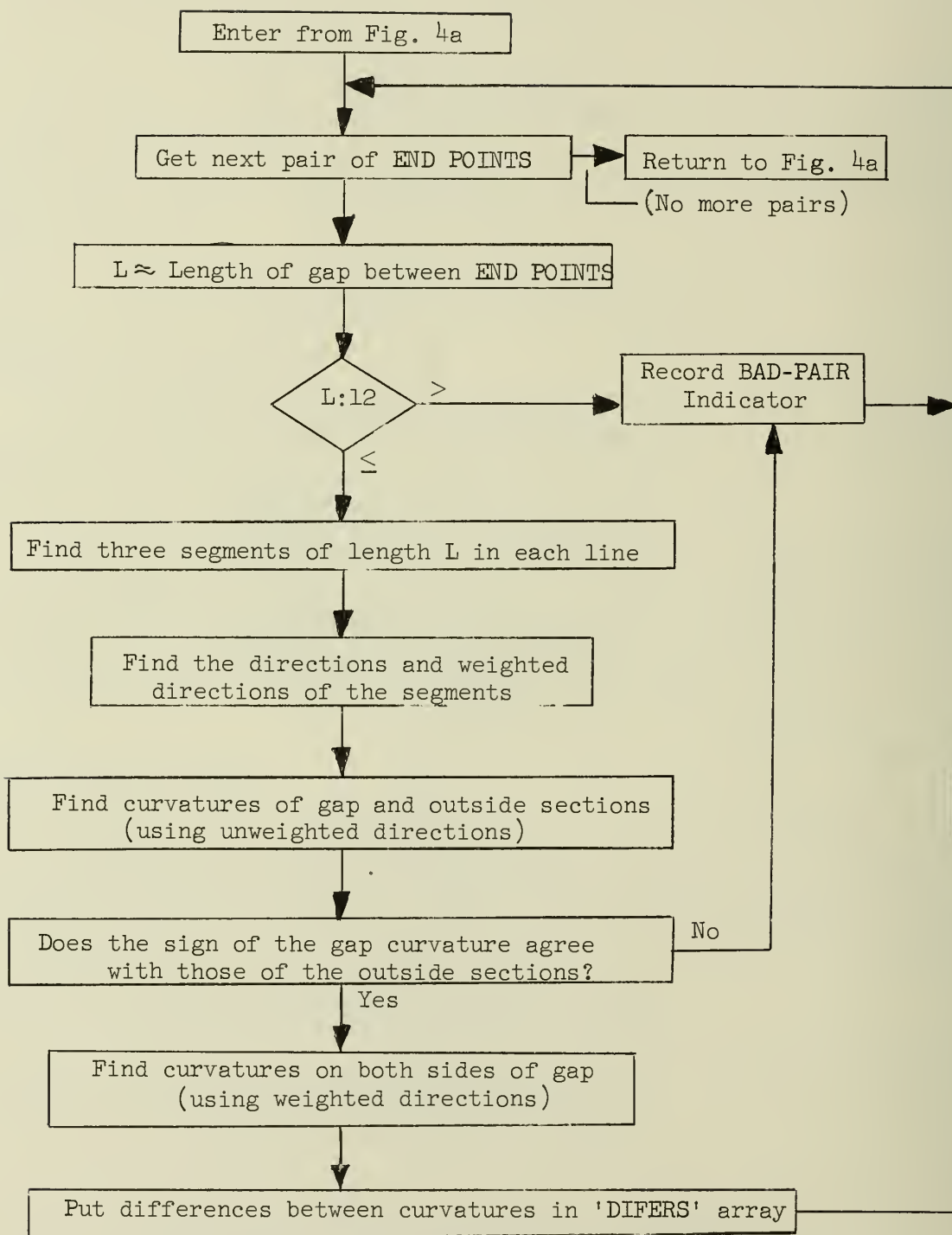


Figure 4b. END POINT Test Flow Chart

For each pair of end points, the length of the gap which may separate them is approximated as the larger of the x or y differences in coordinates. If this length, L, is greater than l2, the gap is too large for the size of the burst being used, and the next set of points is started.

If $L \leq l2$, the subroutine SPREAD is used to find line segments of length L which extend from the end points. This is done three times for each end point. The second segment begins at the far end of the first segment, and the third at the end of the second. This produces three sequential segments of each line (seven in all, including the gap) of length L. The eight coordinate sets of the junction and terminal points are put in a block called SEGSC (segment coordinates).

The SPREAD routine is entered with the starting coordinates in location ENDC and L in PAX index register 8. SPREAD returns to the main program with the other end point of the segment in ENDC.

Then, for each segment, the direction of the line is found by computing $DIR = \text{ARCTANGENT}(DY/DX)$. This direction is changed to conform to our labeling convention by making $1.0 \leq DIR < 9.0$.

With seven segments, there are six junction points (three on each side of the gap). At each junction point the change in direction, DELTAD, is computed as the difference between the two segment directions. This gives an indication of the curvature of the line at that point.

Also, a weighted set of changes in direction are computed by multiplying the DELTAD's nearest the center of the string by 3, multiplying the next nearest DELTAD's by 1.5, and by leaving the outside DELTAD's unmodified. These weighted values are put in a block called WATDD. This gives more emphasis to the curvatures near the gap, and is useful in getting an accurate indication of the continuity of the curvature.

A preliminary test of the signs of the curvatures is made. Three curvatures are computed: one is the sum of the two center DELTAD's, the other two are the sums of the extreme DELTAD's. If the signs of these two outside curvatures agree, the center (or gap) curvature is tested. If its sign disagrees with the other two, computation is terminated and the next pair of end points is started. If the signs of the two outside curvatures disagree, no conclusions can be drawn, and the computation continues.

Two more curvatures are computed. They are the sums of the first three and last three WATDD's. The difference between these two is put in a two-dimensional array called DIFERS. $DIFERS(a,b)$ and $DIFERS(b,a)$ are set to this difference for the end points a and b. If the computation terminates prematurely ($L > 12$, sign test fails, etc.) these DIFERS are set to 10.0 (larger than any possible difference).

This completes the computations for each pair of end points. When all pairs have been computed, the test for the best connections begins.

First, the value 10.0 is stored in all primary diagonal elements of DIFERS. Then, for each value of a, the minimum value of $DIFERS(a,b)$ is found and this b is the best connection for a. The end points a and b are entered in the reconnection list, OKCS (O.K. connections). If no best connection is found, no points are entered in OKCS.

When all a's have been checked, each pair of points in OKCS is connected in NOISY and this becomes the output picture.

4.4 Results

The results of the program are shown in Figure 5. Three sizes of burst were tried, and the outputs show that the errors in reconnection increase as the size of the burst increases.

However, these results show that the routine is sound in principle and can be used in the cleaning of burst noise of small size.



No. 1 PROCESSED

No. 2 ORIGINAL



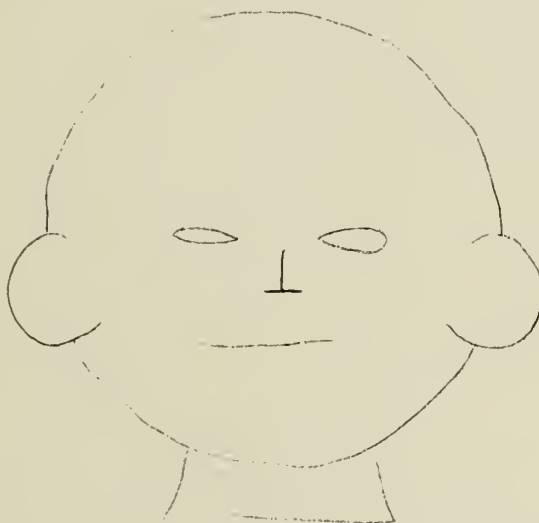
No. 2 PROCESSED

5.1 General

This section outlines a preliminary approach to the extraction of features from a line-like drawing. The program has not been written, but would rely on the reconnection process of Part 4. The purpose of the program will be only a low-level extraction, with no identification of the extracted portions. (Further processing would require the identification.)

5.2 Extraction Process





As an example of this extraction process, consider the face shown below:



For the extraction, it is desirable to separate the ears, eyes, nose, mouth and neck from the outline of the face.

The eyes, nose and mouth are disjointed from the other parts and are easily extracted. The neck and ears are more difficult because they are directly connected to the face outline.

The first step in extracting the features is to label the points of discontinuity ("punctuation" points) as follows:

- node - lines enter the point from four directions; i.e. 
- branch - lines enter the point from three directions; i.e. 
- vertex - lines enter the point from two nonopposite directions, i.e. 
- terminal - only one line enters the point; i.e. 

Directional continuity through punctuation points is then tested. This results in the extraction of the neck and the ears, but leaves the outline of the face in two segments. The reconnection routine is then used to correct such errors.

In more difficult cases, higher level tests will be necessary. These could include a combination of the direction continuity test and the reconnection probability, or could make use of a dictionary of basic forms found in the picture [1].

[1] Ledley, R. S. and J. B. Wilson, "Concept Analysis by Syntax Processing." Proc. of Amer. Doc. Inst., Annual Meeting, Vol. 1, pp. 1-8 (October, 1964).

An addition to PAX has been used in these programs. The addition is called PLOTXY. Given a series of coordinates, this routine plots through the series by connecting the coordinates with straight line segments.

The routine is called by the order:

PLOTXY PLANE,PX,FIRSTC,CX,N,NX

where PLANE,PX is the plane (indexed) where the line is to be plotted,
FIRST,CX is the location of the first coordinate (indexed) to be plotted,
and N,NX is the number of coordinate sets (indexed) in the list.

The previous contents of the plane are undisturbed.

LABEL SUBROUTINE - PAGE 1 OF 1 -

```

LABEL  INDEX  SET,1,4
      BCOLOP  NEWVAL,,CLEAR
LOOPA  INDEX   DECR,1,1
      BOOLOP  TEMP1,,EQUAL,INPUT
      BCOLOP  TEMP2,,EQUAL,INPUT
      SHIFTD  TEMP1,,1,1,2
      SHIFTD  TEMP2,,4,1,2
      BCOLOP  TEMP1,,AND,TEMP1,,TEMP2
      BOOLOP  IODIR,1,AND,TEMP1,,INPUT
      BOOLOP  IODIR+4*144,1,EQUAL,IODIR,1
      BCOLOP  NEWVAL,,OR,NEWVAL,,IODIR,1
      JUMP    NOZERO,LOOPA,1
      INDEX   SET,1,4
      BOOLOP  NOCON,,EQUAL,NEWVAL,,,,BAR
LOOPB  INDEX   DECR,1,1
      INDEX   TRANS,2,1
      INDEX   TRANS,3,1
      INDEX   TRANS,4,1
      INDEX   INCR,2,4
      INDEX   INCR,3,3
      INDEX   DECR,4,1
      TMARK   DL1,1,TEMP1,,NOCON,,IODIR,2,EQ,1
      INDEX   SET,11,3
LOOPB1 INDEX   DECR,11,1
      JUMP    LESS,GOON1A,3,8
      INDEX   DECR,3,8
GOON1A JUMP    MORE,GOON1B,4,-1
      INDEX   INCR,4,8
GOON1B JUMP    LESS,GOON1C,4,8
      INDEX   DECR,4,8
GOON1C TMARK   DL1,3,TEMP2,,TEMP1,,IODIR,4,EQ,1
GOON2  BOOLOP  NEWVAL,,OR,NEWVAL,,TEMP2
      BOOLOP  NEWVAL,,OR,NEWVAL,,TEMP2
      BCOLOP  IODIR,1,OR,IODIR,1,TEMP2
      BCOLOP  IODIR,3,OR,IODIR,3,TEMP2
      INDEX   INCR,3,1
      INDEX   INCR,4,1
      JUMP    NOZERO,LOOPB1,11
      JUMP    NOZERO,LOOPB,1
      BOOLOP  NOCON,,EQUAL,NEWVAL,,,,BAR
      JUMP    RETURN,,8
CLA    DL      2,1,5
      DL      2,2,6
      DL      2,3,7
      DL      2,4,8

```

```

      SETMOD      72
      BCOLOP      ALL,,CLEAR,,,,,BAR
NEWPIC READ      ORIG
      LETTER      ( X)
      PRINT       ORIG,,72,72
      COMMENT     (-ORIGINAL PICTURE)
      TMARK       DLALL,,INPUT,,ALL,,ORIG,,GE,5      LABEL T5
      JUMP        LINK,LABEL,8
      INDEX       SET,1,4
      BCOLOP      T5ALL,,EQUAL,NEWVAL
      BOOLOP      T5BAR,,EQUAL,NOCON
LOOP1  INDEX      DECR,1,1
      BCOLOP      T5,1,OR,IODIR,1,IODIR+4*144,1
      JUMP        NOZERO,LOOP1,1
      TMARK       DLALL,,INPUT,,ALL,,ORIG,,GE,4      LABEL T4
      JUMP        LINK,LABEL,8
      INDEX       SET,1,4
LOOP2  INDEX      DECR,1,1
      BOOLOP      T4,1,OR,IODIR,1,IODIR+4*144,1
      JUMP        NOZERO,LOOP2,1
EXTEND INDEX      SET,1,8      BEGIN EXTENSION OF T5 INTO T4
      INDEX       SET,2,4
LABLEE INDEX      DECR,1,1      LABEL END POINTS
      INDEX       DECR,2,1
      JUMP        MORE,GOON3,2,-1
      INDEX       INCR,2,4
GOON3  TMARK       DLC,1,TEMP1,,T5,2,T5BAR,,EQ,3
      BCOLOP      TEMP2,,EQUAL,T5ALL
      SHIFTD      TEMP2,,1,1,1
      BCOLOP      DENDS,1,AND,TEMP1,,TEMP2
      JUMP        NOZERO,LABLEE,1
      INDEX       SET,1,8
XLABS  INDEX      DECR,1,1      EXTEND LABELS BY ADDING T4(D-1)+T4(D+1)
      INDEX       TRANS,2,1
      INDEX       TRANS,3,1
      INDEX       DECR,2,1
      INDEX       INCR,3,1
      JUMP        MORE,GOON4A,2,-1
      INDEX       INCR,2,4
GOON4A JUMP        MORE,GOON4B,3,-1
      INDEX       INCR,3,4
GOON4B JUMP        LESS,GOON4C,2,4
      INDEX       DECR,2,4
GOON4C JUMP        LESS,GOON4D,3,4
      INDEX       DECR,3,4
GOON4D BCOLOP      DENDSX,1,OR,DENDS,1,T4,2
      BOOLOP      DENDSX,1,OR,DENDSX,1,T4,3
      JUMP        NOZERO,XLABS,1

```

```

INDEX    SET,1,8
INDEX    SET,2,4
GROW     INDEX    DECR,1,1          EXTEND DENDSX INTO T4
INDEX    DECR,2,1
JUMP     MORE,GOON6,2,-1
INDEX    INCR,2,4
GCON6    CHAIN    DL1,1,NEWPTS,1,DENDSX,1,T4,2
JUMP     NOZERO,GROW,1
INDEX    SET,1,8
INDEX    SET,2,4
COMPAR   INDEX    DECR,1,1          WERE NEW POINTS PICKED UP
INDEX    DECR,2,1
JUMP     MORE,GOON5,2,-1
INDEX    INCR,2,4
GCON5    BCOLOP   COMP,,AND,NEWPTS,1,T5,2,BAR
JUMP     NONULL,REITER,COMP
JUMP     NOZERO,COMPAR,1
LETTER   ( EAIN23+B45-6*.$)        NO, PRINT RESULTS
CPRINT   4,72,72,T5,,T5+144,,T5+2*144,,T5+3*144
COMMEN   (-PROCESSED PICTURE)
JUMP     UC,NEWPIC
REITER   INDEX    SET,1,4          YES, REITERATE EXTENSION
ADDPIS   INDEX    DECR,1,1
BCOLOP   TEMP1,,OR,NEWPTS,1,NEWPTS+4*144,1
BOOLOP   T5,1,OR,15,1,TEMP1
BCOLOP   T5BAR,,AND,T5BAR,,TEMP1,,BAR
JUMP     NOZERO,ADDPIS,1
JUMP     UC,EXTEND
CLC      DL       3,8,1,2
          DL       3,1,2,3
          DL       3,2,3,4
          DL       3,3,4,5
          DL       3,4,5,6
          DL       3,5,6,7
          DL       3,6,7,8
          DL       3,7,8,1

```

```

SETMOD 72
READ ORIG
BCOLOP NOISE,,CLEAR
WRITE NOISE,,21,,48
INDEX SET,12,1
ADDNDS INDEX INCR,12,1 ADD NOISE
JUMP MORE,DONE,12,5 NOT TOO MUCH NOISE
MARK CLALL,,TEMP1,,NOISE
BCOLOP NOISE,,EQUAL,TEMP1
BCOLOP NOISEY,,AND,ORIG,,NOISE,,BAR
INPUT EQU NOISEY USE NOISEY FOR INPUT
JUMP LINK,LABEL,8 LABEL INPUT PICTURE
TEST1 BCOLOP EXTRAS,,AND,NOISEY,,NOCON FIND UNLABELED POINTS
JUMP NULL,GOON3,EXTRAS IF NONE GO ON TO 3
INDEX SET,1,8
EXTLAB INDEX DECR,1,1 EXTEND LABELS FROM
MARK CLC,1,TEMP1,,IODIR,1,EXTRAS NEARBY POINTS
BCOLOP IODIR,1,OR,IODIR,1,TEMP1
BCOLOP NOCON,,AND,NOCON,,TEMP1,,BAR
JUMP NOZERO,EXTLAB,1
JUMP UC,TEST1
GCON3 INDEX SET,1,8
BCOLOP ENDS,,CLEAR
FINDES INDEX DECR,1,1 FIND END POINTS
TMARK CLC,1,TEMP1,,IODIR,1,NOCON,,EQ,3
BCOLOP TEMP2,,EQUAL,IODIR,1
SHIFTD TEMP2,,1,1,1
BCOLOP ENDPTS,1,AND,TEMP1,,TEMP2
BCOLOP ENDS,,OR,ENDS,,ENDPTS,1
JUMP NOZERO,FINDES,1
INDEX SET,1,8
LISTXY ENDS,,ELIST,15 LIST END POINTS
JUMP MORE,GOOF1,21,15 TEST FOR TOO MANY
JUMP LESS,GOOF4,21,2 TOO FEW
PAX 0,7 SET UP PARAMETERS IN PROGRAM
SXA TESTC-2,7
TXI *+1,7,-1
SXD GOON13+1,7
ALS 18
STD COUNT1+1
SUB DECR2
STD COUNT2+1
AXC *+2,7
TRA PAX
INDEX SET,6,0
INDEX SET,7,0
NXTPT INDEX INCR,7,1 DO FOR EVERY PAIR
COUNT1 JUMP LESS,GOON4,7,** ** = NO. OF ENDS
INDEX INCR,6,1
COUNT2 JUMP MORE,ADDLIN,6,** ** = NO. OF ENDS -2
INDEX TRANS,7,6
INDEX INCR,7,1

```

```

GOON4 J7094C  **1
LAC      IR-1+6,1      GET THE PAIR OF COORDINATES
LAC      IR-1+7,2
CLA      ELIST,1      FIND LENGTH = MAX(MAGDX,MAGDY)
STO      SEGCS+3
ARS      18
STO      TEMP1
CLA      ELIST,2
STO      SEGCS+4
ARS      18
SUB      TEMP1
STO      DX
SLW      MAGDX
STZ      TEMP1
STZ      TEMP2
CLA      ELIST,2
STA      TEMP1
CLA      ELIST,1
STA      TEMP2
CLA      TEMP2
SUB      TEMP1
STO      DY
SSP
CAS      MAGDX
TRA      **3
TRA      **2
CLA      MAGDX
STO      IR-1+8      IR8 = L
CAS      FX12
TRA      GOOF2
NOP
AXC      **2,7
TRA      PAX
BCOLOP   NEWVAL,,EQUAL,NOCON,,,BAR
J7094C   **1      SPREAD SEGMENTS OF LENGTH L
CLA      SEGCS+3      INTO LINES FROM ENDS
STO      ENDC
AXC      **2,7
TRA      PAX
JUMP     LINK,SPREAD,9
J7094C   **1
CLA      ENDC
STO      SEGCS+2
AXC      **2,7
TRA      PAX
JUMP     LINK,SPREAD,9
J7094C   **1
CLA      ENDC
STO      SEGCS+1
AXC      **2,7

```

	TRA	PAX	
	JUMP	LINK,SPREAD,9	
	J7094C	*+1	
	CLA	ENDC	
	STO	SEGCS	
	CLA	SEGCS+4	
	STO	ENDC	
	AXC	*+2,7	
	TRA	PAX	
	JUMP	LINK,SPREAD,9	
	J7094C	*+1	
	CLA	ENDC	
	STO	SEGCS+5	
	AXC	*+2,7	
	TRA	PAX	
	JUMP	LINK,SPREAD,9	
	J7094C	*+1	
	CLA	ENDC	
	STO	SEGCS+6	
	AXC	*+2,7	
	TRA	PAX	
	JUMP	LINK,SPREAD,9	
	J7094C	*+1	
	CLA	ENDC	
	STO	SEGCS+7	
	AXT	7,1	
FINCO	CLA	SEGCS+7,1	FIND DIRECTION OF SEGMENTS
	ARS	18	
	STO	TEMP1	
	CLA	SEGCS+8,1	
	ARS	18	
	SUB	TEMP1	
	STO	DX	
	STZ	TEMP2	
	STZ	TEMP1	
	CLA	SEGCS+7,1	
	STA	TEMP1	
	CLA	SEGCS+8,1	
	STA	TEMP2	
	CLA	TEMP2	
	SUB	TEMP1	
	STO	DY	
	CLA	FXTDFL	CONVERT DX AND DY TO FLOATING POINT
	ORS	DX	
	ORS	DY	
	ZAC		
	FAD	DX	
	STO	DX	
	ZAC		
	FAD	DY	

	STO	DY	
	NZT	DX	
	TRA	NODX	
	NZT	DY	
	TRA	NODY	
	CLA	DY	
	FDP	DX	
	STQ	DIR+7,1	
	TSX	ATAN,4	ATAN = SYSTEM ARCTANGENT ROUTINE
	TXH	DIR+7,1	EXITS WITH -PI/2 .LE. THETA .LE. PI/2
	STO	DIR+7,1	DIRECTION = ATAN(DY/DX)
	CLA	DX	THEN CONVERT TO 1.LE. DIR .LT.9
	TMI	NEGDX	
	CLA	DY	
	TMI	NEG DY	
	LDQ	DIR+7,1	
GCON9	FMP	F40VPI	
	FAD	FL1	
	CAS	FL9	
	NOP		
	FSB	FL8	
	CAS	FL1	
	TRA	*+3	
	TRA	*+2	
	FAD	FL8	
GCON8	STO	DIR+7,1	
	TIX	FINDD,1,1	
	AXT	6,1	
CURVS	CLA	DIR+6,1	
	FSB	DIR+7,1	FIND CURVATURES AND WEIGHTED CURVATURES
	STO	DELTAD+6,1	
	SSP		
	CAS	FL4	
	TRA	HIGHDD	
	NOP		
GCON10	CLA	DELTAD+6,1	
	XCA		
	FMP	WATES+6,1	
	STO	WATDD+6,1	
	TIX	CURVS,1,1	
STEST	CLA	DELTAD	TEST CONTINUITY OF
	FAD	DELTAD+1	SIGN OF CURVATURE
	STO	COMMON	
	CLA	DELTAD+2	
	FAD	DELTAD+3	
	STO	COMMON+1	
	CLA	DELTAD+4	
	FAD	DELTAD+5	
	TPL	PTEST	
	CLA	COMMON	

	TPL	GOON11	
	CLA	COMMON+1	
	TPL	GOOF2	
SCON11	ZAC		
	AXT	3,1	FIND AVERAGE EFFECTIVE CURVATURE AS
	FAD	WATDD+3,1	SUM OF WEIGHTED CHANGES
	TIX	*-1,1,1	IN DIRECTION
	STO	DDAV1	
	ZAC		
	AXT	3,1	
	FAD	WATDD+6,1	
	TIX	*-1,1,1	
	FSB	DDAV1	FIND DIFFERENCE IN EFFECTIVE
	SLW	DDD	CURVATURES AND PUT IN
	TSX	SETXS,4	'DIFERS' ARRAY FOR FUTURE TEST
	CLA	DDD	
	STO	DIFERS,7	
	STO	DIFERS,6	
	AXC	*+2,7	
	TRA	PAX	
	JUMP	UC,NXTPT	
PTEST	CLA	COMMON	
	TMI	GOON11	
	CLA	COMMON+1	
	TPL	GOON11	
	TRA	GOOF2	
NODX	CLA	DY	
	TMI	D7	
	CLA	FL3	
	TRA	GOON8	
D7	CLA	FL7	
	TRA	GCON8	
NODY	CLA	DX	
	TMI	D5	
	CLA	FL1	
	TRA	GOON8	
D5	CLA	FL5	
	TRA	GCON8	
NEGDx	CLA	DIR+7,1	
	FAD	PI	
	XCA		
	TRA	GOON9	
NEGDY	CLA	DIR+7,1	
	FAD	TWOPI	
	XCA		
	TRA	GOON9	
HIGHED	CLA	DEL TAD+6,1	
	TMI	*+3	
	FSB	FL8	
	TRA	*+2	

	FAD	FL8	
	STO	DELTAD+6,1	
	TRA	GOON10	
ADDLIN	J7094C	**1	PREPARE TO ADD LINES TO ORIG
	AXT	0,1	
	CLA	FL10	STORE ERROR INDICATOR (10.0) ON PRIMARY
	AXT	15,2	DIAGONAL OF DIFERS ARRAY
	STO	DIFERS,1	
	TXI	**1,1,16	
	TIX	**2,2,1	
	AXT	0,5	
	AXT	0,1	
NXTPNT	PXA	0,1	
	XCA		
	MPY	FX15	
	XCA		
	PAX	0,2	
	CLA	FL9P6	
TESTC	CLA	DIFERS,2	TEST FOR BEST CONNECTION
	CAS	MIND	
	TRA	GOON12	
	NOP		
	STO	MIND	
	SXA	BESTC,4	
GOON12	TXI	**1,4,1	
	TXI	**1,2,1	
	TIX	TESTC,3,1	
	CLA	MIND	
	CAS	FL9P3	
	TRA	GOON13	NO POSSIBILITIES
	NOP		
	LAC	BESTC,6	ENTER IN LIST OF O.K. COORDINATES
	CLA	ELIST,6	
	STO	OKCS,5	
	TXI	**1,5,-1	
	SXA	BESTC,1	
	LAC	BESTC,6	
	CLA	ELIST,6	
	STO	OKCS,5	
	TXI	**1,5,-1	
GOON13	TXI	**1,1,1	
	TXL	NXTPNT,1,**	** = ENDPTS - 1
	PCA	0,5	
	TZE	NOCONN	IF NO CONNECTIONS ARE FOUND
	ARS	1	
	STO	IR-1+2	
	AXC	**2,7	
	TRA	PAX	
	INDEX	SET,1,0	
ADDPTS	PLOTXY	NOISEY,,OKCS,1,2	ADD POINTS TO RECONNECT

```

INDEX   DECR,2,1
INDEX   INCR,1,2
JUMP    MORE,ADDPTS,2,0
LETTER  ( X)
PRINT   NOISEY,,72,72
JUMP    UC,ADDNOS
SETXS   LDQ    IR-1+6          SETS 7094 INDEX REGISTERS TO INDEX
      MPY     FX15             2 - DIMENSIONAL DIFERS ARRAY
      XCA
      ADD     IR-1+7
      PAX     0,7
      LDQ     IR-1+7
      MPY     FX15
      XCA
      ADD     IR-1+6
      PAX     0,6
      TRA     1,4
CONE    HALT   DUMP           ** AT LAST **
SPREAD  INDEX  TRANS,1,8
      BCOLOP  END,,CLEAR
      WRITEP  END,,ENDC
      BCOLOP  NEWVAL,,AND,NEWVAL,,END,,BAR
GRCW    INDEX  DECR,1,1      GROW L POINTS BACK ALONG LINE
      MARK    DLSUR,,1EMP1,,END,,NEWVAL
      BCOLOP  NEWVAL,,AND,NEWVAL,,TEMP1,,BAR
      BCOLOP  END,,EQUAL,TEMP1
      JUMP    NOZERO,GROW,1
TESTES  LISTXY  END,,ENDC,2
      JUMP    EXACT,GOOF3,21,0  EXIT IF WOULDN'T GROW
      JUMP    LESS,GOON7,21,3   TEST FOR MULTIPLE ENDS
      BOOFUN  END,,END,,END,,(13+15+17+35+37+57$)  REMOVE SOME
      JUMP    UC,TESTES
GOON7   JUMP    RETURN,,9
GOOF1   COMMEN (-.GT. 15 END POINTS)
      JUMP    UC,ADDNOS
GOOF3   J7094C  *+1          PUT 'BAD SET OF ENDS' INDICATOR
GOOF2   TSX     SETXS,4      IN DIFERS ARRAY
      CLA     FL10
      STO     DIFERS,6
      STO     DIFERS,7
      AXC     *+2,7
      TRA     PAX
      JUMP    UC,NXTPT
GOOF4   COMMEN (-.LT. 2 END POINTS)
      JUMP    UC,ADDNOS
NOCCNN  AXC     *+2,7
      TRA     PAX
      COMMEN  (- NO CONNECTIONS)
      JUMP    UC,ADDNOS

```

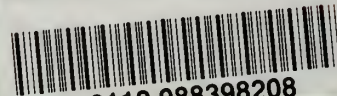
```

*****  CONSTANTS AND BLOCK RESERVATIONS
DLC      DL      3,8,1,2
          DL      3,1,2,3
          DL      3,2,3,4
          DL      3,3,4,5
          DL      3,4,5,6
          DL      3,5,6,7
          DL      3,6,7,8
          DL      3,7,8,1
DECR2    OCT      2000000
FXTOFL   OCT      233000000000
FL1      DEC      1.0
FL3      DEC      3.0
FL4      DEC      4.0
FL5      DEC      5.0
FL7      DEC      7.0
FL8      DEC      8.0
FL9      DEC      9.0
FL9P3    DEC      9.3
FL9P6    DEC      9.6
FL10     DEC      10.0
FX12     DEC      12
FX15     DEC      15
          ASSIGN  DX,DY,MAGDX,MIND,BESTC,DDAV1,DDD
WATES    DEC      1.0,1.5,3.0,3.0,1.5,1.0
DIFERS   RES      15*15
SEGCS    BSS      8
CKCS     BSS      30
COMMON   BSS      3
ELIST    BSS      16
ENDC     BSS      3
DIR      BSS      7
DELTAD   BSS      6
WATDD    BSS      6

```




UNIVERSITY OF ILLINOIS-URBANA
510.84 IL6R no. C002 v.171-187(1965)
Illiac III: a processor of visual inform



3 0112 088398208